UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/805,963	03/22/2004	Kent F. Hayes JR.	RSW920030236US1	2602

45541 7590 08/03/2009

HOFFMAN WARNICK LLC
75 STATE ST
14TH FLOOR
ALBANY, NY 12207

EXAMINER	
WANG, BEN C	

ART UNIT	PAPER NUMBER
2192	

NOTIFICATION DATE	DELIVERY MODE
08/03/2009	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

PTOCommunications@hoffmanwarnick.com

Office Action Summary	Application No.	Applicant(s)	
	10/805,963	HAYES, KENT F.	
	Examiner	Art Unit	
	BEN C. WANG	2192	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *11 May 2009*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-40* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-40* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☐ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. Applicant's response dated May 11, 2009, responding to the Office action mailed February 11, 2009 provided in the rejection of claims 1-40, wherein claims 1, 11, 19, and 31 have been amended.

Claims 1-40 remain pending in the application and which have been fully considered by the examiner.

Applicant's arguments with respect to claims currently amended have been fully considered but are not persuasive. Please see the section of "Response to Arguments" for details.

2. Applicant's amendment necessitated the new ground(s) of rejection presented in this Office action. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a).

Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

Claim Rejections – 35 USC § 102(a)

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102(a) that form the basis for the rejections under this section made in this office action:

> A person shall be entitled to a patent unless –
>
> (a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for a patent.

4. Claims 1-5, 8, 11, 13-15, 18, 19, 21-23, 26, 31, 33-35, and 37-38 are rejected under 35 U.S.C. 102(a) as being anticipated by Ruiz et al. (*A Framework for Resolution of Deployment Dependencies in Java-Enabled Service Gateways, FIDJI 2003*) (hereinafter 'Ruiz')

5. **As to claim 1** (Currently Amended), Ruiz discloses a computer-implemented method for resolving prerequisites for resolving prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework (e.g., Abstract - … This framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies …; Sec. 4 Resolution of Dependencies in Deployment (Requirement)), comprising:

- packaging a native application for a client device and corresponding dependency information within a first OSGi bundle on a server, wherein the corresponding dependency information specifies at least one

prerequisite on which the native application depends for proper operation on the client device (e.g., P. 4, 2nd full Para - … package manager that handle software packages described by meta-data in order to install and configure the software …; Sec. 4 Resolution of Dependencies in Deployment (Requirements), 1st bullet; 4th bullet – Heterogeneity: the dependencies go from OSGi service to service implementation; from service to native libraries, and from native libraries (packages or programs) to other ones; 6th bullet; 10th bullet – Dependencies: different types of dependencies will lead to different activities within the resolution process …)

- polling the client device by the server to determine if the client device has the at least one other prerequisite (e.g., Abstract - … This framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies …; P. 4, 4th point – Find dependencies: … In case of finding it, the delivery process is done recursively …;

  NOTE: *Ruiz* teaches the prerequisite resolution process can be targeted (polled) at either central location or the platform location. Further, *Ruiz* specifically discloses "… A key element in the resolution process is the capability to know if a certain bundle or library has already been deployed on the platform, and its current version (and other configuration data). The resolution process must access this information, either by querying a central configuration repository (that can also been located on the platform

or in another location), or by checking the actual deployment of the library ..." (e.g., second full bullet on page 7 – emphasis added) The platform above referred by *Ruiz* is the OSGi platform device – see first full bullet on page 7 - … This means that the resolution process must not only handle the dependencies and installation process at the OSGI platform … by the operating system supporting the OSGi software platform; in our case it is Linux-Debian – emphasis added));

- obtaining the at least one prerequisite if the client device does not have the at least one prerequisite (e.g., P. 4, 4th point - … If the required component is not available, delivery finishes); and
- loading, by the server, the at least one prerequisite and the native application on the client device (e.g., P.4, 7th point – Installation; P. 9, Sec. of 'Loader' – is the piece of the architecture in charge of downloading the specific elements to be deployed in the platform …)

6. **As to claim 2** (Original) (incorporating the rejection in claim 1), Ruiz discloses the method, further comprising registering the packaged native application and first OSGi bundle after the packaging step, wherein the registering step comprises storing the corresponding dependency information (e.g., P. 4, 8th point – Update register, when a new component is installed it is essential to save it in a database …; 1st full Para - … the component has available meta-data to solve dependencies and conflicts)

7. **As to claim 3** (Original) (incorporating the rejection in claim 1), Ruiz discloses the method, wherein the polling step comprises: identifying the at least one prerequisite to the client device; and receiving a response from the client device, wherein the response indicates whether the client device has the at least one prerequisite (e.g., Abstract - … This framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies …; P. 4, 4th point – Find dependencies: … In case of finding it, the delivery process is done recursively …)

8. **As to claim 4** (Original) (incorporating the rejection in claim 1), Ruiz discloses the method, further comprising: determining the at least one prerequisite, prior to the packaging step; and generating the corresponding dependency information based on the at least one prerequisite (e.g., P. 4, 3rd point – check dependencies is in charge of analyzing the component dependencies …; Sec. 4 Resolution of Dependencies in Deployment (Requirements))

9. **As to claim 5** (Original) (incorporating the rejection in claim 1), Ruiz discloses the method, wherein the at least one prerequisite comprises another native application (e.g., Sec. 5 Resolution of Dependencies in Deployment (Architectural Design), Native Resolver - … The native resolver is thus a wrapper to the Debina resolver (apt-dpkg) …)

10. **As to claim 8** (Previously Presented) (incorporating the rejection in claim 1), Ruiz discloses the method wherein the method is performed recursively for the at least one prerequisite to resolve prerequisites for the at least one prerequisite (e.g., Abstract - … This framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies …; P. 4, 4th point – Find dependencies: … In case of finding it, the delivery process is done recursively …)

11. **As to claim 11** (Currently Amended), Ruiz discloses a computer-implemented method for resolving prerequisites for native applications, comprising:

- packaging a native application for a client device and corresponding dependency information within a first bundle on a server, wherein the dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (e.g., ., P. 4, 2nd full Para - … package manager that handle software packages described by meta-data in order to install and configure the software …; Sec. 4 Resolution of Dependencies in Deployment (Requirements), 1st bullet; 4th bullet – Heterogeneity: the dependencies go from OSGi service to service implementation; from service to native libraries, and from native libraries (packages or programs) to other

ones; 6th bullet; 10th bullet – Dependencies: different types of dependencies will lead to different activities within the resolution process …)

- polling the client device to determine if the client device has the at least one other prerequisite (e.g., Abstract - … This framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies …; P. 4, 4th point – Find dependencies: … In case of finding it, the delivery process is done recursively …;
NOTE: *Ruiz* teaches the prerequisite resolution process can be targeted (polled) at either central location or the platform location. Further, *Ruiz* specifically discloses "… A key element in the resolution process is the capability to know if a certain bundle or library has already been deployed on the platform, and its current version (and other configuration data). The resolution process must access this information, either by querying a central configuration repository (that can also been located on the platform or in another location), or by checking the actual deployment of the library ..." (e.g., second full bullet on page 7 – emphasis added) The platform above referred by *Ruiz* is the OSGi platform device – see first full bullet on page 7 - … This means that the resolution process must not only handle the dependencies and installation process at the OSGI platform … by the

operating system supporting the OSGi software platform; in our case it is Linux-Debian – emphasis added));

- obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server (e.g., P. 4, 4th point - … If the required component is not available, delivery finishes); and
- installing, by the server, the first bundle and the second bundle within an environment of the client device (e.g., P.4, 7th point – Installation; P. 9, Sec. of 'Loader' – is the piece of the architecture in charge of downloading the specific elements to be deployed in the platform …)

12. **As to claims 13-15**, please refer to claims **3-5** as set for the above accordingly.

13. **As to claim 18** (Previously Presented) (incorporating the rejection in claim 11), please refer to claim **8** as set forth above accordingly.

14. **As to claim 19** (Currently Amended), Ruiz discloses a computerized system for resolving prerequisites for native applications, comprising:

- a packaging system for packaging a native application for a client device and corresponding dependency information within a first bundle on a server, wherein the dependency information specifies at least one

prerequisite on which the native application depends for proper operation on the client device (e.g., P. 4, 2nd full Para - … package manager that handle software packages described by meta-data in order to install and configure the software …; Sec. 4 Resolution of Dependencies in Deployment (Requirements), 1st bullet; 4th bullet – Heterogeneity: the dependencies go from OSGi service to service implementation; from service to native libraries, and from native libraries (packages or programs) to other ones; 6th bullet; 10th bullet – Dependencies: different types of dependencies will lead to different activities within the resolution process …);

- a communication system for polling the client device to determine if the client device has the at least one other prerequisite (e.g., Abstract - … This framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies …; P. 4, 4th point – Find dependencies: … In case of finding it, the delivery process is done recursively …;
NOTE: *Ruiz* teaches the prerequisite resolution process can be targeted (polled) at either central location or the platform location. Further, *Ruiz* specifically discloses "… A key element in the resolution process is the capability to know if a certain bundle or library has already been deployed on the platform, and its current version (and other configuration data). The resolution process must access this information, either by querying a central configuration repository (that can also been located on the platform

or in another location), or by checking the actual deployment of the library ..." (e.g., second full bullet on page 7 – emphasis added) The platform above referred by *Ruiz* is the OSGi platform device – see first full bullet on page 7 - … This means that the resolution process must not only handle the dependencies and installation process at the OSGI platform … by the operating system supporting the OSGi software platform; in our case it is Linux-Debian – emphasis added));

- a resolution system for obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to the server (e.g., P. 4, 4th point - … If the required component is not available, delivery finishes); and
- a bundle loading system for loading, by the server, the first OSGi bundle and the second OSGi bundle on the client device (e.g., P.4, 7th point – Installation; P. 9, Sec. of 'Loader' – is the piece of the architecture in charge of downloading the specific elements to be deployed in the platform …)

15. **As to claim 21** (Original) (incorporating the rejection in claim 19), please refer to claim **3** above, accordingly.

16. **As to claim 22** (Original) (incorporating the rejection in claim 19), please refer to claim **4** above, accordingly.

17. **As to claim 23** (Original) (incorporating the rejection in claim 19), please refer to claim **5** above, accordingly.

18. **As to claim 26** (Original) (incorporating the rejection in claim 19), please refer to claim **3** above, accordingly.

19. **As to claim 31** (Currently Amended), Ruiz discloses a program product stored on a recordable medium for resolving prerequisites for native applications, which when executed, comprises:

- program code for packaging a native application for a client device and corresponding dependency information within a first bundle on a server, wherein the dependency information specifies at least one prerequisite on which the native application depends for proper operation on the client device (e.g., P. 4, 2nd full Para - … package manager that handle software packages described by meta-data in order to install and configure the software …; Sec. 4 Resolution of Dependencies in Deployment (Requirements), 1st bullet; 4th bullet – Heterogeneity: the dependencies go from OSGi service to service implementation; from service to native libraries, and from native libraries (packages or programs) to other ones; 6th bullet; 10th bullet – Dependencies: different types of dependencies will lead to different activities within the resolution process …)

- program code for polling the client device to determine if the client device has the at least one other prerequisite (e.g., Abstract - … This framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies …; P. 4, 4th point – Find dependencies: … In case of finding it, the delivery process is done recursively …;

  NOTE: *Ruiz* teaches the prerequisite resolution process can be targeted (polled) at either central location or the platform location. Further, *Ruiz* specifically discloses "… A key element in the resolution process is the capability to know if a certain bundle or library has already been deployed on the platform, and its current version (and other configuration data). The resolution process must access this information, either by querying a central configuration repository (that can also been located on the platform or in another location), or by checking the actual deployment of the library ..." (e.g., second full bullet on page 7 – emphasis added) The platform above referred by *Ruiz* is the OSGi platform device – see first full bullet on page 7 - … This means that the resolution process must not only handle the dependencies and installation process at the OSGI platform … by the operating system supporting the OSGi software platform; in our case it is Linux-Debian – emphasis added));

- program code for obtaining the at least one prerequisite if the client device does not have the at least one prerequisite, wherein the at least one prerequisite is packaged within a second bundle that is accessible to

the server (e.g., P. 4, 4th point - … If the required component is not available, delivery finishes); and

- program code for loading, by the server, the first bundle and the second bundle on the client device (e.g., P.4, 7th point – Installation; P. 9, Sec. of 'Loader' – is the piece of the architecture in charge of downloading the specific elements to be deployed in the platform …)

20. **As to claim 33** (Original) (incorporating the rejection in claim 31), please refer to claim **3** above, accordingly.

21. **As to claim 34** (Original) (incorporating the rejection in claim 31), please refer to claim **4** above, accordingly.

22. **As to claim 35** (Original) (incorporating the rejection in claim 31), please refer to claim **5** above, accordingly.

23. **As to claim 37** (Original) (incorporating the rejection in claim 31), Ruiz discloses the program product, wherein the at least one prerequisite is packaged with corresponding dependency information within the second OSGi bundle (e.g., P. 4, 3rd point – check dependencies is in charge of analyzing the component dependencies …; Sec. 4 Resolution of Dependencies in Deployment (Requirements))

24. **As to claim 38** (Original) (incorporating the rejection in claim 31), Ruiz discloses the program product, wherein the client device includes: program code for determining whether the client device has the at least one prerequisite; and program code for generating and sending a response to the server (e.g., Abstract - … This framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies …; P. 4, 4th point – Find dependencies: … In case of finding it, the delivery process is done recursively …)

Claim Rejections – 35 USC § 103(a)

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

25. Claims 10, 27-28, 30, and 40 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ruiz in view of Hall et al., (*Component Deployment on OSGi: The Gravity Case, January 29, 2003, Fractal Workshop – LSR-Adele*) (hereinafter 'Hall')

26. **As to claim 10** (Original) (incorporating the rejection in claim 1), further, Ruiz discloses the framework performs the recursive identification and resolution

of dependencies between services in OSGi platforms, including the resolution of native dependencies (e.g., Abstract) but does not explicitly disclose the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle.

However, in an analogous art of *component deployment on OSGi: the gravity case*, Hall discloses the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle (e.g., Slide 7 – Bundle Manifest Example - Import-Package, specification-version)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Hall into the Ruiz's system to further provide the method, the system, and the program product, wherein a name and version of the native application is represented in a name and version of the OSGi bundle.

The motivation is that it would enhance the Ruiz's system by taking, advancing and/or incorporating the Hall's system which provides the framework of a factory service concept built on top of OSGi and further standardizes OSGi component creation as once suggested by Hall (e.g., Slides 30-31, 35-41 – Extended OSGi Component Model for Gravity)

27. **As to claim 27** (Original) (incorporating the rejection in claim 19), Hall discloses the system, wherein the dependency information specifies an identity

and a version of the at least one prerequisite required by the native application (e.g., Slide 7 – Bundle Manifest Example - Import-Package, specification-version)

28. **As to claim 28** (Original) (incorporating the rejection in claim 27), please refer to claim **5** above, accordingly.

29. **As to claim 30** (Original) (incorporating the rejection in claim 19), please refer to claim **10** as set forth above accordingly.

30. **As to claim 40** (Original) (incorporating the rejection in claim 31), please refer to claim **10** as set forth above accordingly.

31. Claims 6-7, 9, 12, 16-17, 20, 24-25, 29, 32, 36, and 39 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ruiz in view of Liang et al. (*Bundle Dependency in Open Services Gateway Initiative Framework Initialization, 2002, IEEE*) (hereinafter 'Liang')

32. **As to claim 6** (Original) (incorporating the rejection in claim 1), Further, Ruiz discloses the framework performs the recursive identification and resolution of dependencies between services in OSGi platforms, including the resolution of native dependencies (e.g., Abstract) but does not explicitly disclose other limitations stated below.

However, in an analogous art of *Bundle Dependency in Open Services Gateway Initiative Framework Initialization*, Liang discloses the method, wherein the at least one prerequisite is packaged with corresponding dependency information within a second OSGi bundle, and wherein the obtaining step comprises obtaining the second OSGi bundle (e.g., Fig. 3 – Bundle Dependency Relationship; Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para)

Therefore, it would have been obvious to one of ordinary skill in the art, at the time the invention was made to combine the teachings of Liang into the Ruiz's system to further resolve prerequisites for native applications in an Open Service Gateway Initiative (OSGi) framework.

The motivation is that it would enhance the Ruiz's system by taking, advancing and/or incorporating Liang's system which provides the framework can look up all events of those bundles to manage the bundle dependency automatically as once suggested by Liang (e.g., sec. of e) A New OSGi Component Model, 1st Para., Lines 13-16)

33. **As to claim 7** (Original) (incorporating the rejection in claim 6), Liang discloses the method, wherein loading step comprises:

- installing the first OSGi bundle and the second OSGi bundle within an OSGi environment of the client device (e.g., Sec. 1 of Introduction, 2nd Para., Lines 13—20; Sec. of Bundle Dependency During Framework Initialization, 3rd Para.);

- deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (e.g., Sec. of Introduction, 2nd Para., Lines 1-12); and
- removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10)

34. **As to claims 9** (Original) (incorporating the rejection in claim 1), Liang discloses the method and the program product, wherein the dependency information is expressed as a package import statement (e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para., Lines 5-11 – import-package field in its manifest file)

35. **As to claim 12** (Original) (incorporating the rejection in claim 11), Liang discloses the method, wherein the first OSGi bundle and the second OSGi bundle are registered on the server after being packaged with the first native application and the at least one prerequisite (e.g., Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para., Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para., Lines 2-12)

36. **As to claim 16** (Original) (incorporating the rejection in claim 11), Liang discloses the method, further comprising:

- deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (e.g., Sec. of Introduction, 2nd Para., Lines 1-12); and
- removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10)

37. **As to claim 17** (Original) (incorporating the rejection in claim 11), Liang discloses the method, wherein the at least one prerequisite is packaged with corresponding dependency information within the second OSGi bundle (e.g., Fig. 3 – Bundle Dependency Relationship; Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 3rd Para.)

38. **As to claim 20** (Original) (incorporating the rejection in claim 19), Liang discloses the system, wherein packaging system further registers the first OSGi bundle after being packaged with the first native application (e.g., Sec. 1, 1st Para., Lines 12-14 – open services include service discovery, service registration, service deployment, service processing, and service security, 2nd Para, Lines 17-19 – from a shared service registry; Sec. of d) By Using ServicEvent and Event Handling Mechanism of OSGi, 1st Para, Lines 2-12)

39. **As to claim 24** (Original) (incorporating the rejection in claim 19), Liang discloses the system, wherein bundle loading system comprises:

- an export system for installing the first OSGi bundle and the second OSGi bundle within the OSGi environment of the client device (e.g., Sec. 1 of Introduction, 2nd Para., Lines 13—20; Sec. of Bundle Dependency During Framework Initialization, 3rd Para.);
- a deployment system for deploying the first OSGi bundle and the second OSGi bundle within a native environment of the client device (e.g., Sec. of Introduction, 2nd Para., Lines 1-12); and
- a removal system for removing the native application from within the first OSGi bundle and the at least one prerequisite from within the second OSGi bundle (e.g., Sec. of c) Let The Third Party Bundle To Manage the Service Dependency In a Centralized Control Way, 2nd Para., Lines 8-10)

40. **As to claim 25** (Original) (incorporating the rejection in claim 19), please refer to claim **17** above, accordingly.

41. **As to claim 29** (Previously Presented) (incorporating the rejection in claim 19), please refer to claim **9** above, accordingly.

42. **As to claim 32** (Original) (incorporating the rejection in claim 31), please refer to claim **20** above, accordingly.

43. **As to claim 36** (Original) (incorporating the rejection in claim 31), please refer to claim **24** above, accordingly.

44. **As to claim 39** (Original) (incorporating the rejection in claim 31), please refer to claim **9** as set forth above accordingly.

Response to Arguments

45. Applicant's arguments filed on May 11, 2009 have been fully considered but they are not persuasive.

In the remarks, Applicant argues that, for examples:

(A.1) *Ruiz* fails to disclose polling the client device by the server to determine if the client device has the at least one other prerequisite (stated in first paragraph on page 13 – emphasis added)

Examiner's response:

(R.1) As per argument one (A.1) above, *Ruiz* teaches the (prerequisite) resolution process can be targeted (polled) at either central location or the platform location. Further, *Ruiz* specifically discloses "… A key element in the resolution process is the capability to know if a certain bundle or library has already been deployed on the platform, and its current version (and other

configuration data). The resolution process must access this information, either by querying a central configuration repository (that can also been located on the platform or in another location), or by checking the actual deployment of the library ..." (e.g., second full bullet on page 7 – emphasis added) The platform above referred by *Ruiz* is the OSGi platform device – see first full bullet on page 7 - … This means that the resolution process must not only handle the dependencies and installation process at the OSGI platform … by the operating system supporting the OSGi software platform; in our case it is Linux-Debian – emphasis added)

Conclusion

46. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Ben C. Wang whose telephone number is 571-270-1240. The examiner can normally be reached on Monday - Friday, 8:00 a.m. - 5:00 p.m., EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Tuan Q. Dam can be reached on 571-272-3695. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Ben C Wang/
Ben C. Wang
Examiner, Art Unit 2192

/J. Derek Rutten/
Primary Examiner, Art Unit 2192